SASL / SSL / TLS

Położenie: (nie dotyczy)

Po co TLS/SSL?

SASL (*Simple Authentication and Security Layer*) - znany także pod nazwą *SMTP AUTH*, zdefiniowany w RFC 2222, zabezpiecza tylko uwierzytelnianie, nie szyfruje przesyłanych wiadomości. Polega na tym, że przy wysyłaniu poczty podaje się login i hasło (domyślnie są wysyłane jako *plain-text*). Znajduje się w pakiecie *Cyrus SASL*.

Problem braku szyfrowania przesyłanych wiadomości rozwiązuje *SSL* oraz *TLS* (nowsze rozwiązanie), które szyfruje dane już w momencie przesyłania hasła. Aby możliwe było zastosowanie *SSL*, serwer, na którym jest *Apache*, musi mieć stały numer IP (także w przypadku wykupienia usługi hostingowej z *cPanel*, potrzebne jest wykupienie stałego numeru IP). Możliwe wtedy staje się zastosowanie tzw. *self-signed certificates* (certyfikat bezpłatny). Można także wykupić certyfikację w profesjonalnej firmie.

Rodzaje szyfrowania

RSA (*Rivest-Shamir-Adleman*) - asymetryczny system kryptograficzny opracowany w 1977 chroniony patentem przez *RSA Security*.
DES (*Data Encryption Standard*) - algorytm szyfrowania opracowany przez *IBM* i uznany w 1977 za oficjalny standard w USA. W 1997 został złamany po 90 dniach starań (14000 komputerów). Zastąpiono więc ten algorytm Triple-DES (potrójny DES).
MD5 (*Message Digest Algorithm*) - opisany w RFC1321, 128-bitowy. Do tej pory niezłamany.
SHA (*Secure Hash Algorithm*) - opracowany w 1993 roku przez *National Institute of Standards and Technology*, 160-bitowy.

Podstawowe pojęcia

SSL (*Secure Socket Layer*) - powstał w 1994 w *Netscape*. Rok później wychodzi SSLv3. Protokół SSLv2 ma duże wady, gdyż można na nim wymusić stosowanie obniżonego szyfrowania (np. 40-bitowego). Należy więc wyłączyć w przeglądarce protokół SSLv2. Komunikacja SMTP poprzez SSL przebiega na porcie 465, a komunikacja HTTPS na porcie 443. W roku 2014 wykryto dziurę w protokole, która umożliwia przechwycenie danych za pomocą ataku POODLE (jest to nieusuwalna wada samego protokołu), więc największe firmy informatyczne zalecają jego wyłączenie. Aby sprawdzić, czy nasza przeglądarka jest podatna na atak, należy wejść na stronę *https://www.poodletest.com*.
TLS (*Transport Layer Security*) - powstaje w 1999 roku (zdefiniowany w RFC 2246) i jest zaakceptowany przez *Netscape* i *Microsoft*. Jest równoważny SSL3. Można skorzystać z tego na serwerze Postfix pod warunkiem zainstalowania *OpenSSL*. Komunikacja SMTP za pomocą TLS przebiega po portach 587. TLS różni się od SSL tym, że oferuje klientom tylko możliwość szyfrowania (na żądanie, ustala to z nimi), podczas gdy SSL inicjuje szyfrowanie połączenia od razu zanim cokolwiek innego prześle.
STARTTLS – jest to rozszerzenie, które oferuje komunikacji opartej o *plain-text* przejście na wyższy poziom zabezpieczenia (szyfrowanie, ale nadal na porcie 25 – co jest rozwiązaniem wygodniejszym dla różnych klientów), zamiast utworzenia osobnego kanału szyfrowania TLS (port 587, w przypadku zdefiniowania „*submission*” w pliku „*master.cf*”). Innymi słowy, STARTTLS to szyfrowana komunikacja na nieszyfrowanym kanale, a SSL/TLS to nieszyfrowana komunikacja na szyfrowanym kanale. Serwer informuje klienta pocztowego, że przejście na szyfrowanie TLS jest możliwe, jednak nie wymusza na kliencie tego szyfrowania (jedynie oferuje je jako możliwość). Gdy klient odpowie „*STARTTLS*”, serwer prześle komunikat „*220 Ready to start TLS*” (lub komunikat błędu).

Zastosowanie TLS z Postfiksem

W *Postfix* jest możliwe zastosowanie szyfrowanej komunikacji pocztowej poprzez tzw. system TLS oparty o klucze (prywatny i publiczny). Certyfikat może być zaoferowany klientowi przez serwer,

ale także sam klient może zaoferować swój certyfikat serwerowi w celu potwierdzenia swojej tożsamości.
Instalujemy pakiet:
emerge openssl

Prywatna autoryzacja certyfikatów

Zazwyczaj certyfikaty są autoryzowane przez publiczne instytucje CA (*Certification Authority*) i jest to opcja płatna. Możemy jednak zrobić to sami za pomocą skryptu perlowego, tzn. utworzyć zaszyfrowany główny klucz prywatny CA (*/home/robert/demoCA/private/cakey.pem*) oraz główny publiczny certyfikat CA (*/home/robert/demoCA/cacert.pem*):
$ **/etc/ssl/misc/CA.pl -newca**
W wyniku działania tego skryptu powstaną następujące pliki i foldery:

- *demoCA/cacert.pem* – certyfikat główny CA
- *demoCA/private/cakey.pem* – główny prywatny klucz CA
- *demoCA/index.txt* – baza danych certyfikatów podpisanych przez główny certyfikat CA
- *demoCA/serial* – zawiera następny numer seryjny do użycia jako ASCII text
- *demoCA/newcerts* – kopia wszystkich certyfikatów podpisanych przez główny certyfikat CA; certyfikaty będą mieć w nazwie numer seryjny
- *demoCA/newcerts/01.pem* – kopia pierwszego podpisanego certyfikatu
- *demoCA/crl/crl.pem* – lista unieważnionych certyfikatów.

Uwaga: Jako że skrypt zawiera błąd (nie tworzy pliku „*serial*” lecz „*crlnumber*”), tuż po uruchomieniu skryptu należy na innej konsoli wykonać polecenie:
$ **ln -sv /home/robert/demoCA/crlnumber /home/robert/demoCA/serial**

Info: „*PEM pass phrase*” to hasło potrzebne do zaszyfrowania pliku klucza (serwis poprosi później o to hasło, gdy zostanie np. zrestartowany).

Generujemy prywatny klucz na koncie zwykłego użytkownika (nie jest zaszyfrowany, ale powinien być do odczytu tylko przez właściciela) oraz certyfikat (*certificate request*). Certyfikat i prywatny klucz mogą znajdować się w jednym pliku lub w dwóch osobnych):
$ **(umask 077; openssl req -new -newkey rsa:2048 -nodes -keyout 3bird-key.pem -out 3bird-req.pem)**

Aby certyfikat spełniał swoje zadanie musi być podpisany przez kogoś, komu ufamy (jakieś CA). Ale możemy także (w ramach oszczędności finansowych) podpisać go sami za pomocą naszego prywatnego klucza (czyli podpisujemy sami sobie):
$ **openssl ca -out 3bird-cert.pem -days 365 -infiles 3bird-req.pem**

Uwaga: W przypadku przeglądarki internetowej, po przejściu na stronę szyfrowaną nie pojawi się żaden komunikat, jeśli przeglądarka będzie znała CA, które podpisało certyfikat. Jeśli nie będzie znała (np. będzie to certyfikat podpisany przez nas samych) – poprosi o jego akceptację i potwierdzenie umieszczenia go na liście zaufanych certyfikatów.

Tworzenie certyfikatów – metoda 2

Generowanie klucza prywatnego zabezpieczonego hasłem (nikomu nie ujawniamy!):
$ **openssl genrsa -des3 -out kluczPrywatny.pem 2048**
Generowanie pliku „żądania certyfikatu” (w czasie tworzenia podajemy różne dane o naszej firmie / witrynie, bez polskich znaków):
$ **openssl req -new -key kluczPrywatny.pem -out plikZadaniaCertyfikatu.pem**
Tak wygenerowany plik „żądania certyfikatu”, można przesłać do publicznej instytucji CA w celu jego potwierdzenia.
Weryfikujemy klucz prywatny:
$ **openssl rsa -noout -text -in kluczPrywatny.pem**
Weryfikujemy plik „żądania certyfikatu”:
$ **openssl req -noout -text -in plikZadaniaCertyfikatu.pem**

Testowanie TLSa

Najpierw sprawdzamy, jakie porty są w ogóle otwarte i czy nie są blokowane przez firewalla:

 # nmap nazwaSerwera

W pliku konfiguracyjnym postfiksa */etc/postfix/main.cf*:
smtpd_tls_auth_only = yes (autoryzację SASL opartą o *plain-text*, przeprowadź dopiero po nawiązaniu szyfrowanego połączenia TLS → a więc nie pojawi się już w sesji telnetowej wyrażenie *AUTH*)

Testujemy działanie TLS (nie używamy telneta):
openssl s_client -starttls smtp -crlf -connect server.3bird:25
lub
openssl s_client -crlf -connect server.3bird:465 (aby sprawdzić działanie SSL)
...
ehlo client.3bird
(powinno pojawić się napis STARTTLS)

STARTTLS (wydajemy polecenie)
220 Ready to start TLS

Opis sesji

Klient	***Serwer***
Hello Serwer! Możemy porozmawiać używając: Wersja: TSLv1; jeśli go nie znasz to SSLv3. Wymiana klucza: RSA; jeśli go nie znasz to Diffie-Hellman. Metoda szyfrowania tajnego klucza: Potrójny DES; jeśli nie to DES. Hash (skrót wiadomości): SHA-1; jeśli go nie znasz to MD5. Metoda kompresji danych: PKZIP; jeśli nie, to gzip. Losowy numer: 196201083.	*Słyszę cię. Porozmawiajmy używając:* Wersja: TLSv1. Wymiana klucza: RSA. Metoda szyfrowania tajnego klucza: DES. Hash (skrót wiadomości): SHA-1. Metoda kompresji danych: PKZIP. Losowy numer: 823455127.

Po udzieleniu odpowiedzi klientowi na *Hello*, serwer wysyła swój certyfikat. Certyfikat serwera podpisany jest przez zaufany ośrodek jakim może być urząd ds. certyfikatów, czyli CA (*Certification Authority*). Klient weryfikuje certyfikat serwera za pomocą klucza publicznego udostępnionego przez CA. Następnie serwer może zażądać certyfikatu od klienta (choć nie jest to konieczne). Nawet jeśli to zrobi, klient nie musi go wysyłać (po pierwsze dlatego, że nie ma komercyjnego certyfikatu; po drugie, i tak będzie weryfikowany np. za pomocą karty kredytowej). Przesyłane wiadomości i tak są szyfrowane.

Szyfrowanie plików

OpenSSL może być wykorzystane także do szyfrowania pojedynczych plików:
openssl enc -aes-256-cbc -in plaintext.file -out cyphertext.file (kodowanie)
openssl enc -aes-256-cbc -d -in cyphertext.file > plaintext.file (odkodowanie)

Ostatnia aktualizacja: 5 września 2016.